Hunting for binary trees in binary character sets: efficient algorithms for extraction, enumeration and optimization

by

David Bryant
Department of Mathematics and Statistics,
University of Canterbury, Christchurch, New Zealand

No. 124 April, 1995

Hunting for binary trees in binary character sets: efficient algorithms for extraction, enumeration and optimization

David Bryant*

Department of Mathematics & Statistics
University of Canterbury, Christchurch,
New Zealand

April 5, 1995

Abstract

We describe an efficient solution to a useful variant of the NP-hard Maximum Compatible Subset problem. Let S be a collection of binary characters. We first wish to determine whether there exists a subcollection $S' \subseteq S$ such that S' equals the set of splits of a binary tree. We provide a polynomial time algorithm that solves this problem and also counts how many such subcollections there are. Furthermore, if each of the given characters is weighted, we can efficiently find the subcollection of characters that corresponds to a binary tree and has maximum or minimum summed weight. The algorithm can be extended to deal with subcollections $S' \subset S$ that correspond to trees with bounded vertex degree, and still have polynomial time efficiency.

Keywords: Character compatibility; binary tree; phylogenetic trees; phylogenetic inference.

*Email: djb@math.canterbury.ac.nz

1 Introduction

A **binary character** is a function from the set of taxa to the set $\{0, 1\}$, usually representing some property that distinguishes one group of taxa from the others. Binary characters can be obtained, for example, from genetic data as in Section 4, or from morphological distinctions like vertebrate/invertebrate. Each binary character induces a unique **split** of the set of taxa, that is, a partition into two groups comprising those assigned a 1 and those assigned a 0. We represent a split by an unordered pair $\{A, A'\}$, where A and A' are disjoint sets whose union is the set of taxa. Sets of binary characters or splits are one of the most common starting points for phylogenetic analysis.

A **phylogenetic tree** is a tree with no vertices of degree two, and with each leaf labelled by a unique member of the set of taxa. The split $\{A, A'\}$ **fits** a phylogenetic tree T if we can remove an edge of T to give two subtrees, one with leaf set A and the other with leaf set A'. The **set of splits of a tree** T is the set of splits that fit T. Hence a given set of characters can sometimes be represented by one phylogenetic tree, in which case we say that the characters are **compatible**.

But how does one analyse phylogenetic information from a set of characters that do not fit directly into a phylogenetic tree? One standard approach is to find a largest subset of characters that is compatible. However this Maximum Compatible Subset problem was shown to be NP-hard by Day and Sankoff [6]. The related problem for weighted characters – the Optimal Weighted Character Subset problem (determine a subset of compatible characters that optimizes the sum of weights) – is therefore also NP-hard. In fact, problems over the whole discipline of phylogenetics exhibit a discouraging tendency to be NP-hard.

One property of NP-completeness is that minor restrictions on the parameters in the input can lead to problems that can be solved in polynomial time [8]. Two notable recent examples are the Maximal Agreement Subtree problem and the Perfect Phylogeny problem. The first is NP-hard, [3], but can be solved efficiently when one of the input trees has bounded vertex degree [3, 7]; The second is NP-complete [4, 12], but can be solved efficiently when the number of character states is fixed [2, 10]. The problems we consider here are essentially restricted versions of the Maximal Compatible Subset problem and the Optimal Weighted Character Subset problem. Instead of looking for arbitrary subsets of compatible characters we limit our attention to those subsets of characters that correspond to binary trees. This is not an artificial restriction, but reflects the prominence of binary trees in phylogenetics. We show, furthermore, that weakening this restriction to cover trees with bounded vertex degree gives a problem that can still be solved efficiently.

2 A special case: sets of clusters

We first present our algorithm not in terms of collections of characters but in terms of clusters and rooted phylogenetic trees. A **cluster** is just a subset of the set of taxa. In a **rooted** phylogenetic tree, one internal vertex is distinguished and called the **root**. In a binary rooted tree all internal vertices have degree 3, except the root which has degree 2. Given two vertices u and v in a rooted phylogenetic tree we say that u is a **descendent** of v if the path from u to the root passes through v. A cluster A **fits** a rooted tree T if there is some internal vertex v such that A equals the set of leaves of T that are descendents of v. The **set of clusters of a tree** T equals the set of clusters that fit T.

We can now state the problem formally.

INSTANCE: Collection C of subsets of a leaf set X such that $X \in C$.
PROBLEM: Determine if there is a subcollection $C' \subseteq C$ such that C' equals the set of clusters of some binary rooted phylogenetic tree with leaf set X. If one or more such subcollections exist then print one out.

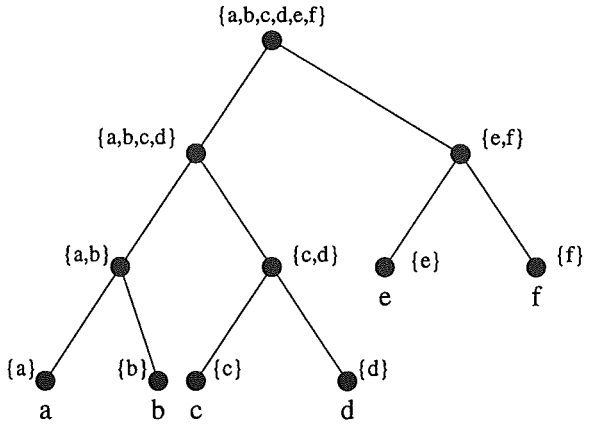

Figure 1 : A rooted binary phylogenetic tree with leaf set $\{a, b, c, d, e, f\}$.

Consider the tree in Figure 1. The internal vertices are labelled with their respective clusters. The leaf set $\{a, b, c, d, e, f\}$ is the union of two disjoint clusters, $\{a, b, c, d\}$ and $\{e, f\}$, which in turn equal the union of two disjoint clusters and so on, until we reach the leaves of the tree.

A given collection, C, of clusters contains a subcollection corresponding to a binary tree if and only if the cluster X ($\in C$) can be expressed as the union of two disjoint clusters in C, which in turn can each be expressed as the union of two disjoint clusters in C and so on, right down to the level of singleton clusters.

2.1 Principal algorithm

Let k be the number of clusters in C and n be the number of taxon in X. We assume that C contains all the single element clusters. In practice these can simply be appended to the input data.

> (1) Sort the collection C such that $C_i \subset C_j$ implies $i < j$. Remove duplicate clusters. Write the resulting set as $\{C_1, C_2, \ldots, C_K\}$, where K is the number of distinct clusters in C

One suitable sorting scheme is to (i) represent each cluster in C by an n-digit binary number, with a 1 for the ith digit if the ith taxon is in the cluster, and a 0 otherwise, and (ii) sort the clusters into ascending order of their associated binary numbers.

> (2) For each $i = 1, \ldots, K$, create a list $D[i]$ of unordered pairs $\{p, q\}$ such that $C_p \cap C_q = \emptyset$ and $C_p \cup C_q = C_i$.

The table D can be constructed by simply considering all $\binom{K}{2}$ pairs of clusters in C, checking if they intersect and if their union is in C. Note that for each i, $D[i]$ contains fewer than $K/2$ pairs.

> (3) For each i, calculate $m[i]$ which equals the number of binary trees with leaf set C_i and clusters contained in C. Remove from $D[i]$ all pairs $\{p, q\}$ such that $m[p] = 0$ or $m[q] = 0$.

To calculate the $m[i]$'s, consider the clusters of C in order from C_1 to C_K. For each cluster C_i:

- If C_i is a singleton then $m[i] := 1$.
- else if $D[i]$ is empty, then $m[i] := 0$
- else $m[i] := \sum_{\{p,q\} \in D[i]} m[p] \times m[q]$

> (4) The total number of binary trees with leaf set X and clusters in C equals $m[K]$, since $C_K = X$.

Step (1) can be completed in $O(nk)$ time using radix sort [9, 1], where n is the number of taxa and k is the number of splits. For arbitrary $\{p, q\}$ calculating $C_p \cap C_q$ and $C_p \cup C_q$ takes $O(n)$ time. Since C is ordered it takes $O(\log K)$ time to determine whether $C_p \cup C_q$ is in C, which is just $O(n)$ time since $K \leq 2^n$. Hence step (2) can be completed in $O(nK^2)$ time, where K is the number of *distinct* splits in C. Finally, evaluating $m[i]$ for all i takes at most $O(K^2)$ steps. Therefore steps (1), (2), (3) and (4) take $O(nk + nK^2)$ time. Note that K is often significantly smaller that k.

Correctness of this method is given by the following theorem, proved in Section 5.

Theorem 1 *After the completion of the above algorithm, $m[i]$ equals the number of distinct rooted binary trees with clusters in C_i and leaf set X.*

(5) It is now a simple matter to list the subcollections C' that correspond to binary trees. Given a collection of clusters C, we say that a particular cluster $C_i \in C$ is **unresolved** if there is no pair of clusters $C_p, C_q \in C$ such that $C_p \cap C_q = \emptyset$ and $C_p \cup C_q = C_i$. The following recursive procedure, GETSUBCOLLECTIONS, builds up a list of clusters. At each level it chooses from the list an unresolved cluster C_i and adds to the list two new disjoint clusters C_p and C_q such that $C_p \cup C_q = C_i$.

The first parameter is the list being built up, initially just $\{C_K\}$, and the second parameter is the table D constructed in step (2) and pruned in step (3).

```
Procedure GETSUBCOLLECTIONS(C',D)

if the only unresolved clusters in C' are singletons then
   output the subcollection C'
else
   choose any unresolved cluster C_i ∈ C' that is not a singleton
   for {p,q} in D[i]
      GETSUBCOLLECTIONS(C' ∪ {C_p, C_q}, D)
   end (for)
end (if)
end.
```

If C' is the set of clusters of some tree T, then the graph of T can be easily retrieved directly from C'.

Note that it is not always practical to list all of the subcollections of C that correspond to binary trees, as the following theorem illustrates.

Theorem 2 *Given a set X with n taxa, we can construct a collection C with $(n^2+n)/2$ clusters so that the number of subcollections $C' \subseteq C$ that correspond to rooted binary trees with leaf set X exceeds 2^{n-2}.*

(Proof in Section 5)

2.2 Optimization

We can use the technique for counting trees to solve the following, seemingly more difficult, problem.

INSTANCE: Collection C of subsets of a finite set X. Weighting function $w : C \to \Re$.
PROBLEM: Find a subcollection $C' \subseteq C$ that maximizes $\sum_{C_i \in C'} w(C_i)$ such that C' equals the set of clusters of some rooted binary tree.

The following procedure, MAXWEIGHTTREE, takes three parameters: The first parameter is the collection C; The second parameter is the table D constructed in step (2) and pruned in step (3); The third is the table m constructed in step (3).

The procedure returns two tables, M and D^*. For each i, $M[i]$ is the weight of the maximum weight subcollection that corresponds to a binary tree with leaf set C_i. The table D^* is a reduced version of D used to list the maximum weight subcollections.

```
Procedure MAXWEIGHTTREE(C, D, m)

for i = 1 to K
   if m[i] ≠ 0 then
      if C_i is a singleton then M[i] := w(C_i)
      else
         choose {p, q} in D[i] that maximizes M[p] + M[q]
         M[i] := M[p] + M[q] + w(C_i)
         D*[i] := {{r, s} : M[r] + M[s] + w(C_i) = M[i]}
      end (if)
   end (if)
end (for)
end.
```

The procedure has complexity $O(K^2)$, where K is the number of distinct clusters in C. To list all the optimal subcollections, execute GETSUBCOLLECTIONS($\{C_K\}, D^*$) . There are sometimes, however, an exponentially large number of optimal subcollections. Correctness is given by the following theorem, proved in Section 5.

Theorem 3 *After execution of the procedure* MAXWEIGHTTREE, *if* $m[i] \neq 0$ *then* $M[i]$ *equals the weight of the maximum weight binary tree with leaf set* C_i *and clusters in* C.

Only a slight modification is needed to solve the corresponding minimization problem.

2.3 Extension to non-binary trees

In step (2) of the algorithm a list $D[i]$ is created for each $i = 1, \ldots, K$. The list contains those pairs $\{p, q\}$ such that $C_p \cap C_q = \emptyset$ and $C_p \cup C_q = C_i$. The list $D[i]$ can be extended to include triples $\{p_1, p_2, p_3\}$ or d-tuples $\{p_1, p_2, \ldots, p_d\}$ such that $C_{p_1}, C_{p_2}, \ldots, C_{p_d}$ are pairwise disjoint and $C_{p_1} \cup C_{p_2} \cup \cdots \cup C_{p_d} = C_i$. This suggests a solution to the following problem:

INSTANCE: Collection C of subsets of a leaf set X such that $X \in C$. Number $d \geq 2$.
PROBLEM: Determine if there is a subcollection $C' \subseteq C$ such that C' equals the set of clusters of some rooted phylogenetic tree with leaf set X and with each vertex having no more than d adjacent descendent vertices. If one of more such subcollections exist then print one out.

We outline of the extended algorithm.

> (1) Sort the collection C such that $C_i \subset C_j$ implies $i < j$. Remove duplicate clusters. Write the resulting set as $\{C_1, C_2, \ldots, C_K\}$, where K is the number of distinct clusters in C
>
> (2) For each $i = 1, \ldots, K$, create a list $D[i]$ of tuples $\{p_1, \ldots, p_j\}$ with at least two and at most d elements, such that $C_{p_1}, C_{p_2}, \ldots, C_{p_j}$ are pairwise disjoint and $C_{p_1} \cup C_{p_2} \cup \cdots \cup C_{p_j} = C_i$.

The table D can be constructed by simply considering all $\binom{K}{2} + \binom{K}{3} + \cdots + \binom{K}{d}$ tuples of clusters in C, where each tuple has at least two and at most d elements. Check if the clusters in each tuple are disjoint and if their union is in C. Note that for each i, $D[i]$ contains fewer than K^{d-1} tuples.

> (3) For each i, calculate $m[i]$ which equals the number of phylogenetic trees with leaf set C_i and clusters contained in C, and with each vertex having no more than d adjacent descendent vertices. Remove from $D[i]$ all tuples $\{p_1, \ldots, p_j\}$ such that $m[p_k] = 0$ for some $k = 1, \ldots, j$.

To calculate the $m[i]$'s, consider the clusters of C in order from C_1 to C_K. For each cluster C_i:

- If C_i is a singleton then $m[i] := 1$.
- else if $D[i]$ is empty, then $m[i] := 0$

- else

$$m[i] := \sum_{\{p_1,p_2,\ldots,p_j\}\in D[i]} m[p_1] \times m[p_2] \times \cdots \times m[p_j].$$

(4) The total number of binary trees with leaf set X and clusters in C, and with each vertex having no more than d adjacent descendent vertices, equals $m[K]$.

Let n by the number of taxa, k the number of clusters and K the number of distinct clusters. Step (1) takes $O(nk)$ time, as before. Calculating $C_{p_1} \cap \cdots \cap C_{p_d}$ and $C_{p_1} \cup \cdots \cup C_{p_d}$ for arbitrary $\{p_1, \ldots, p_d\}$ takes $O(nd)$ time. Hence step (2) can be completed in $O(ndK^d)$ time. Evaluating $m[i]$ for all i takes at most $O(K^d)$ time. Therefore steps (1), (2) and (3) take $O(nk + ndK^d)$ time.

The procedures GETSUBCOLLECTIONS and MAXWEIGHTTREE are both easily extended and still take only polynomial time.

3 Sets of characters

We now return to considering sets of splits (binary characters). Formally stated, our two main problems are:

INSTANCE: Collection S of splits on leaf set X.
PROBLEM: Determine if there is a subcollection $S' \subseteq S$ such that S' equals the set of splits of some unrooted binary phylogenetic tree. If one or more such subcollections exist then print one out.

INSTANCE: Collection S of splits on leaf set X. Weighting function $w : C \rightarrow \Re$.
PROBLEM: Find a subcollection $S' \subseteq S$ that maximizes $\sum_{S_i \in S'} w(S_i)$ such that S' equals the set of splits of some unrooted binary phylogenetic tree.

Both of these problems can be transformed into cluster problems.

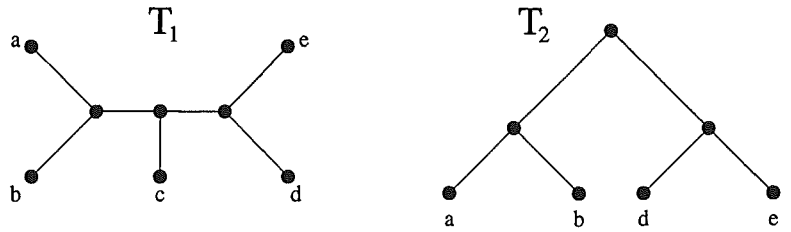

Figure 2 : An unrooted tree T_1 with corresponding rooted tree T_2.

Consider the two trees T_1 and T_2 in Figure 2. Tree T_1 has splits

$$\{\{a, bcde\}, \{b, acde\}, \{c, abde\}, \{d, abce\}, \{e, abcd\}, \{ab, cde\}, \{de, abc\}\}.$$

The tree T_2 is obtained from the tree T_1 by rooting T_1 at the internal vertex adjacent to c, and then removing the leaf c together with its adjacent edge. Because of this relationship, it is possible to calculate the clusters of T_2 directly from the splits of T_1. For each split $\{A, A'\}$ of T_1, if $c \in A$, then A' is a cluster of T_2, otherwise A is a cluster of T_2. Hence the clusters of T_2 are $\{\{a\}, \{b\}, \{a, b, d, e\}, \{e\}, \{a, b\}, \{d, e\}\}$. We utilize this relationship to transform the above splits problems into cluster problems.

Theorem 4 *Let X be a set of taxa and $c \in X$. Let f_c be the mapping from splits of X to clusters of $X - \{c\}$ given by*

$$f_c(\{A, A'\}) = \begin{cases} A & \text{when } c \in A' \\ A' & \text{when } c \in A \end{cases}$$

Then f_c is a one-to-one mapping with inverse

$$f_c^{-1}(A) = \{A, X - A\}.$$

Given any collection S' of splits of X, the set of clusters $f_c(S')$ equals the set of clusters of some rooted tree if and only if S' equals the set of splits of some unrooted tree.

(Proof in Section 5)

It follows that the number of subcollections $S' \subseteq S$ that equal the set of splits of a binary unrooted phylogenetic tree is the same as the number of subcollections $C' \subseteq f_c(S)$ that equal the set of clusters of a binary rooted phylogenetic tree.

Given a set S of k splits of a set X, choose arbitrary $c \in X$ and calculate $f_c(S)$. This can be done in $O(nk)$ time. Apply the above cluster algorithms to $f_c(S)$ to count the total number of subcollections corresponding to binary trees. A particular subcollection $C' \subset f_c(S)$, such as one returned by the algorithm GETSUBCOLLECTIONS, can be transformed into the corresponding set of splits using f_c^{-1}.

If the original set S of splits was weighted then weight the elements of $f_c(S)$ by $w(A) = w(f_c^{-1}(A))$, $A \in f_c(S)$. If C' is an optimal subcollection of $f_c(S)$ then $f_c^{-1}(C')$ will be an optimal subcollection of S.

4 Application: *Xenopus* frogs

As an illustration, we apply the above algorithms to data obtained from the mitochondrial DNA of Xenopus frogs by Carr et al. [5]. We appended the eight trivial characters (those distinguishing a single taxon from the rest), giving a data table with eight taxa and 38 characters, 31 of which are distinct.

There are 41 different binary trees with splits contained in the Xenopus data set. We optimized with respect to two different weighting schemes.

(1) We weighted each distinct character by the number of duplicates of that character in the original data set. There were four such trees, see Figure 3. An exhaustive search using PAUP revealed that these four trees are exactly the trees with minimum length under the parsimony criterion.

(2) We weighted the characters using a technique devised by Lento [11]. Given an unweighted set of splits S, the **Lento weighting** of a split $\{A, A'\}$ in S is the number of splits in S that are not compatible with $\{A, A'\}$. Recall that two splits are compatible if there is an unrooted tree that contains them both. The **Lento score** of a tree is the sum of the Lento weightings of its splits.

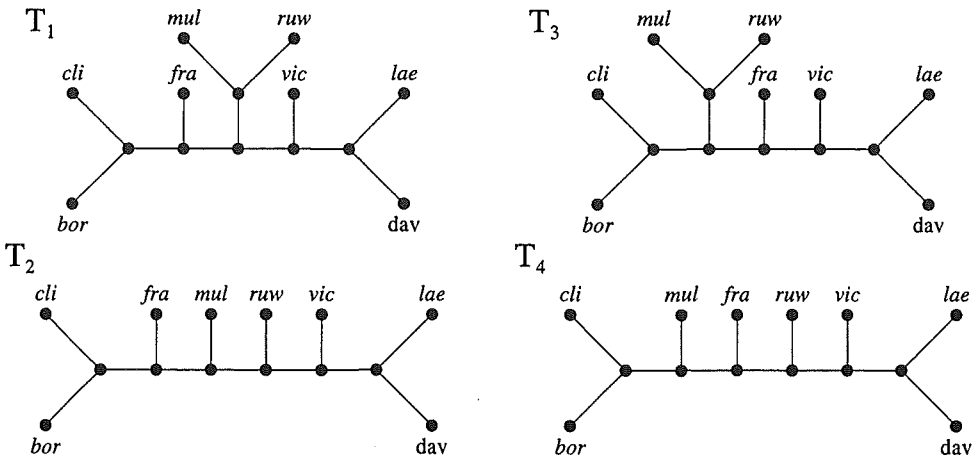

Figure 3 : Four trees relating mtDNA maps for *Xenopus* taxa. The abbreviations for the taxa are: *bor*, borealis; *cli*, clivii; dav, laevis "davis"; *fra*, fraseri; *lae*, l. laevis; *mul*, muelleri; *ruw*, ruwenzoriensis; *vic*, l. victorianus. Tree T_1 has minimum Lento score.

The binary tree with splits in the *Xenopus* data set and minimum Lento score was the tree T_1, one of the four trees constructed in part (1). Note that by using the above algorithms the optimum tree was determined without listing all 41 trees.

5 Proofs

5.1 Proof of Theorem 1

We proceed by induction. The theorem is clearly true when $i = 1$. Suppose that it is true when $i = 1, \ldots, j-1$. Let $\mathcal{T}_j$ be the set of rooted binary trees with leaf set C_j and clusters contained in C. We will show that $m[j] = |\mathcal{T}_j|$.

Let $T \in \mathcal{T}_j$. There are two clusters C_p and C_q in T, and therefore also in C, such that $C_p \cap C_q = \emptyset$ and $C_p \cup C_q = C_j$. It follows that the pair $\{p, q\}$ is contained in $D[j]$. We can partition $\mathcal{T}_j$ so that each block in the partition corresponds to a different pair in $D[j]$. Hence

$$|\mathcal{T}_j| = \sum_{\{p,q\} \in D[j]} (\text{ \# trees in } \mathcal{T}_j \text{ clusters } C_p \text{ and } C_q).$$

Fix any such pair $\{p, q\} \in D[j]$. The number of trees in $\mathcal{T}_j$ that contain both C_p and C_q equals the number of trees with leaf set C_p and clusters contained in C multiplied by the number of trees with leaf set C_q and clusters contained in C. By the induction hypothesis this equals $m[p] \times m[q]$. Therefore,

$$\begin{aligned} |\mathcal{T}_j| &= \sum_{\{p,q\} \in D[j]} m[p] \times m[q] \\ &= m[j] \end{aligned}$$

as required. □

5.2 Proof of Theorem 2

Given n, let $X = \{1, 2, \ldots, n\}$ and construct

$$C_{a,b} = \{x \in X : a \leq x \leq b\}$$

$$C = \{C_{a,b} : a \leq b, \text{ where } a, b \in X\}.$$

Then $|C| = (n^2 + n)/2$. Let $\mathcal{T}_n$ be the set of binary trees with leaf sets X and clusters in C. We claim that $|\mathcal{T}_n| > 2^{n-2}$.

The result is easily verified for $n = 1, 2, 3$. Suppose that $|\mathcal{T}_k| > 2^{k-2}$. Given any tree T in $\mathcal{T}_k$, we construct two new trees.

1. Create a new root with two descendents. Let one descendent be the root of T, and the other descendent be the leaf $(k+1)$.

2. Replace leaf k of T with a new vertex. Let the leaves k and $k+1$ be the two descendents of this new vertex.

Both of these trees are binary and have clusters in C. Hence for every tree in $\mathcal{T}_k$ there are at least two trees in $\mathcal{T}_{k+1}$, all of which are distinct. By the induction hypothesis

$$|\mathcal{T}_{k+1}| \geq 2 \times |\mathcal{T}_k| > 2 \times 2^{k-2} = 2^{(k+1)-2}$$

as required. □

5.3 Proof of Theorem 3

Again we let $\mathcal{T}_i$ denote the set of binary trees with leaf set C_i and clusters in C. Let $w(T)$ denote the sum of the weights of the clusters of a phylogenetic tree T. We need to prove two things:

(i) $w(T) \leq M[i]$ for all $T \in \mathcal{T}_i$.

(ii) There is $T \in \mathcal{T}_i$ such that $w(T) = M[i]$.

We proceed by induction. If $i = 1$ then C_i is a leaf, so (i) and (ii) hold. Suppose that (i) and (ii) are true for $i = 1, \ldots, j-1$.

Let $T \in \mathcal{T}_j$. There is $\{p, q\} \in D[j]$ such that C_p and C_q are clusters of T. Now $C_p \cup C_q = C_j$, so the two clusters correspond to the two subtrees branching off the root of T. Using the induction hypothesis applied to the subtrees of T, we obtain

$$w(T) = M[p] + M[q] + w(C_j) \leq M[j],$$

proving (i).

For (ii), let $\{p, q\}$ be a pair in $D[j]$ such that

$$M[p] + M[q] + w(C_j) = M[j].$$

By the induction hypothesis there is a tree T_p with leaf set C_p and clusters in C, and a tree T_q with leaf set C_q and clusters in C, such that $w(T_p) = M[p]$ and $w(T_q) = M[q]$. Construct a new tree T by connecting a new root r to the root of T_p and also to the root of T_q. Then $T \in \mathcal{T}_j$ and $w(T) = M[j]$. □

5.4 Proof of Theorem 4

It is straightforward to verify that f_c has the given inverse. We prove the second result.

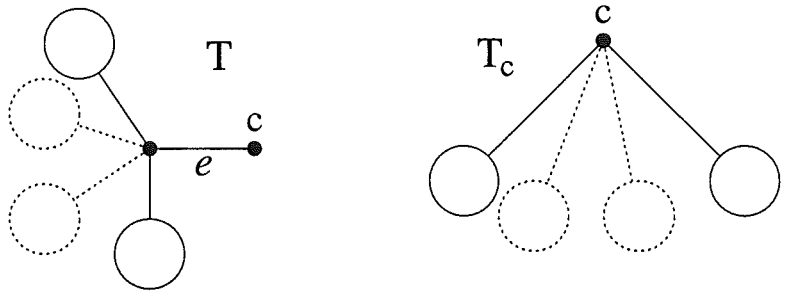

Figure 4 : The transformation from T, on the left, to T_c, on the right. The hollow circles represent subtrees.

Suppose that S' equals the set of splits of an unrooted tree T with leaf set X. Construct the rooted tree T_c from T by (i) making the vertex adjacent to the leaf c the root of T_c and (ii) removing the leaf c and its adjacent edge (see Figure 4). Let e be the edge adjacent to the leaf c in T. The tree T_c equals the subtree of T rooted at the opposite end of e from c. Therefore a set A is a cluster of T_c if and only if $(A, X - A)$ is a split in T. Hence $f_c(S')$ is the set of clusters of some rooted tree.

Conversely, if $f_c(S')$ equals the set of clusters of some rooted tree T_c, then construct T by attaching a leaf c and an edge to the root of T_c. Then $f_c^{-1}(f_c(S')) = S'$ will equal the set of splits of T. □

Acknowledgements

I wish to thank my supervisor, Dr Mike Steel, for his encouragement and his assistance proof-reading.

References

[1] Aho, A.V., Hopcroft, J.E. and Ullman, J.D. 1974. *The Design and Analysis of Computer Algorithms.* Addison-Wesley.

[2] Argarwala, R. and Fernández-Baca, D. 1993. A polynomial-time algorithm for the perfect phylogeny problem when the number of character states are fixed. *Proc. of the 34th annual Symposium on Foundations of Computer Science.*

[3] Amir, A. and Keselman, D. 1994. Maximum agreement subtrees in multiple evolutionary trees. *Proc. of the 35th IEEE Annual Symp. on Foundations of Computer Science*, 758-769.

[4] Bodlaender, H., Fellows, M. and Warnow, T. 1992. Two strikes against perfect phylogeny. *Proc. of the 19th International Congress on Automata, Languages and Programming (ICALP)*, 273-287.

[5] Carr, S.M., Brothers, A.J. and Wilson, A.C. 1987. Evolutionary inferences from restriction maps of mitochondrial DNA from nine taxa of Xenopus frogs. *Evolution* 41(1), 176-188.

[6] Day, W.H.E. and Sankoff, D. 1986. Computational complexity of inferring phylogenies by compatibility. *Syst. Zool.* 35(2), 224-229.

[7] Farach, M., Przytycka, T.M. and Thorup, M. 1995. On the Agreement of many trees. Submitted to *Inf. Proc. Lett.*

[8] Gary, M. R. and Johnson, D.S. 1979. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* W.H. Freeman and Co., San Francisco, CA.

[9] Gusfield, D. 1991. Efficient algorithms for inferring evolutionary trees. *Networks* 21, 19-28.

[10] Kannan, S. and Warnow, T. 1995. A fast algorithm for the computation and enumeration of perfect phylogenies. To appear, *ACM/SIAM Symposium on Discrete Algorithms.*

[11] Lento, G.M., Hickson, R.E., Chambers, G.K. and Penny, D. 1995. Use of spectral analysis to test hypotheses on the origin of pinnepids. *Mol. Biol. Evol.* 12(1), 28-52.

[12] Steel, M.A. 1992. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification* 9, 91-116.

[13] Swofford, D.L. 1985 Phylogenetic analysis using parsimony (PAUP), Illinois Natural History Survey, Champaign, IL.